Exercise : Data security

Purpose:
In this exercise, you will learn how to use the built-in features of Informix that control data security.

Task 1. Using GRANT statements.

In this task, you will be granting table level privileges for a second login on the **items** table. These grant privileges include the ability to create and drop tables, select and insert rows, and update only one column in the **items** table.

1. Open two dbaccess sessions.
2. In the first dbaccess session:
   - Use the GRANT statement to allow all users to connect to your database.

     GRANT CONNECT TO public;
   - Use the GRANT statement to give user **bob** the ability to create a table and drop any objects that he owns, but not the ability to drop the database.
   - **GRANT RESOURCE TO bob;**
   - Exit dbaccess.
3. In a second dbaccess session:
   - Connect to your database as user **bob**.

     Do this by using the **Connection > Connect** options from the dbaccess menu. Select the **dev** server, then log in as user **bob (password bob)**.
   - Create a new table called **bob1** with one column **col1** of data type character(10).

     CREATE TABLE bob1 (

     col1 CHAR(10)
     );
   - Drop the new table.

     DROP TABLE bob1;

   Did this work?

   Yes. Bob was allowed to drop his own table.

- Connect to the **sysmaster** database.
- Drop the **stores_demo** database.

What happened?

The DROP DATABASE command returns the following error:
389: No DBA permission.
User bob does not have sufficient privileges to drop the stores_demo database.

4. In the first session:
   - Reconnect to your **stores_demo** database.
   - Revoke all privileges on the **items** table from public.

     REVOKE ALL ON items FROM PUBLIC;
   - Using the GRANT statement, give user **jane** the ability to select and insert rows in the **items** table.

     GRANT SELECT, INSERT ON items TO jane;
5. In the second session:
   - Connect (**dev**) to your database as user **jane (password jane)**.
   - Delete the **items** from **order_num** 1022**.**

     DELETE FROM items WHERE order_num = 1022;

   Why did this statement fail?

   Because jane does not have delete privileges on the items **table.**
   274: No DELETE permission for items.
   - Select all the **items** for **order_num** 1022.

     SELECT * FROM items WHERE order_num = 1022;

   What happened?

   Jane was able to select the items because she has SELECT permission on the items **table. (No rows found.)**
6. In the first session:
   - Using the GRANT statement, give user **sam** the ability to update only the **manu_code** column in the **items** table.

     GRANT UPDATE (manu_code) ON items TO sam;

7. In the second session:
   - Connect to your database as user **sam (password sam).**
   - Update the **items** table and set the **manu_code** to HSK for order 1001.

     UPDATE items
     SET manu_code = "HSK"
     WHERE order_num = 1001;

   What happened?

   The update failed because sam only has update privileges on the manu_code column and cannot select the order_num column.
   272: No SELECT permission for items.order_num.

Task 2. Using GRANT and REVOKE statements.

In this task, you will revise the privileges previously assigned to the other users using the GRANT and REVOKE statements.

1. In your first session:

   User **jane** no longer needs to select from or insert into the **items** table.
   - Execute the SQL statement needed to change **jane's** access privileges.

     REVOKE SELECT, INSERT ON items FROM jane;

   User **joe** needs to select from only the **order_num** and **total_price** columns of the **items** table.
   - Execute the SQL statement needed to change **joe's** access privileges.

     GRANT SELECT (order_num, total_price) ON items TO joe;

2. In your second session:
   - Connect to your database as user **joe (password joe)**.
   - Select all rows from the **items** table.

     SELECT * FROM items;

   What happened?

   Only the columns granted SELECT privileges (order_num, total_price) are returned to the user.

Task 3. Using roles.

In this task, you will create a role for the purchasing department and grant and revoke privileges for various SQL statements.

1. In your first session:

   - Create a role for the purchasing department.

   CREATE ROLE purchasing;

   - Grant the purchasing department role the appropriate privileges so that users can insert into the **stock** table, but only update the **unit_price** column.

   GRANT INSERT, UPDATE (unit_price) ON stock TO purchasing;

   - Revoke all privileges on the **stock** table from public except SELECT.

   REVOKE ALL ON stock FROM public;

   GRANT SELECT ON stock TO public;

   - Grant the purchasing role to user **frank**.

   GRANT purchasing TO frank;

2. In your second session:

   - Connect to your database as user **frank (password frank)**.
   - Insert a row into the **stock** table using the following statement:

   INSERT INTO stock (stock_num, manu_code)

   VALUES (1, "ANZ");

   Did the insert work?

   No. The insert returns the following error:

   275: The insert privilege is required for this operation.

   What do you need to do to make it work?

   Since no default roles have been defined, user frank needs to grant himself the purchasing role and re-insert the row.

   - Fix the problem and insert the row.

   SET ROLE purchasing;

   INSERT INTO stock (stock_num,manu_code)

   VALUES (1,"ANZ");

3. In the second session:

- Connect to your database as user **mary (password mary)**.
- Insert a row into the **stock** table using the following statement:

```
INSERT INTO stock (stock_num, manu_code)
    VALUES (2, "ANZ");
```

Did the insert work?

No. The insert returns the following error:
275: The insert permission is required for this operation.

What do you need to do to make it work?

Since only the purchasing role can insert into the stock table, user mary needs to grant herself the purchasing role and re-insert the row.

- Fix the problem and insert the row.

```
SET ROLE purchasing;

INSERT INTO stock (stock_num,manu_code)
    VALUES (2,"ANZ")
```

The SET ROLE statement returns the following errors:

19805: No privilege to set to the role.

111: ISAM error: no record found.

Mary has not been granted the role of purchasing, so she cannot assign herself to that role.

Task 4. Using GRANT and REVOKE on fragments.

In this task, you will revoke all on the **customer** table and give select privileges to public. For a second user, you will revoke all on the **customer** table. Grant the second user only insert on the **customer** table in dbspace4.

1. In the first session:
   - Revoke all privileges on the **customer** table from public.

   REVOKE ALL ON customer FROM PUBLIC;

   - Grant SELECT to **public** on the **customer** table.

   GRANT SELECT ON customer TO PUBLIC;

   - Grant DELETE on the **customer** table in **dbspace4** to user **bob**.

   GRANT FRAGMENT DELETE ON customer(dbspace4) TO bob;

2. In the second session:
   - Connect to your database as user **bob (password bob)**.
   - Delete the customer with the last name of "**Currie**" from the **customer** table.

   DELETE FROM customer
   WHERE lname = "Currie";

   Is the row deleted?

   No, because the customer named "Currie" is located in dbspace2 and bob only has delete permissions on customers in dbspace4.

Results:
In this exercise, you learned how to use the built-in features of Informix that control data security.